

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
21 June 2001 (21.06.2001)

PCT

(10) International Publication Number
WO 01/44932 A1

(51) International Patent Classification⁷: **G06F 9/44**

(21) International Application Number: PCT/US00/20723

(22) International Filing Date: 27 July 2000 (27.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/170,958 15 December 1999 (15.12.1999) US

(71) Applicant: **MICROSOFT CORPORATION [US/US];**
One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: **ALI, Arish**; Apt. D102, 4723 148th Avenue
NE, Bellevue, WA 98007 (US). **AUST, Brian, S.**; 6110

140th Court N.E., Redmond, WA 98052 (US). **HELLYAR, Paul, S.**; 304 10th Street, Unit 3, Kirkland, WA 98033 (US). **GROSSMAN, Joel, K.**; 13782 NE 60th Way #123, Redmond, WA 98052 (US).

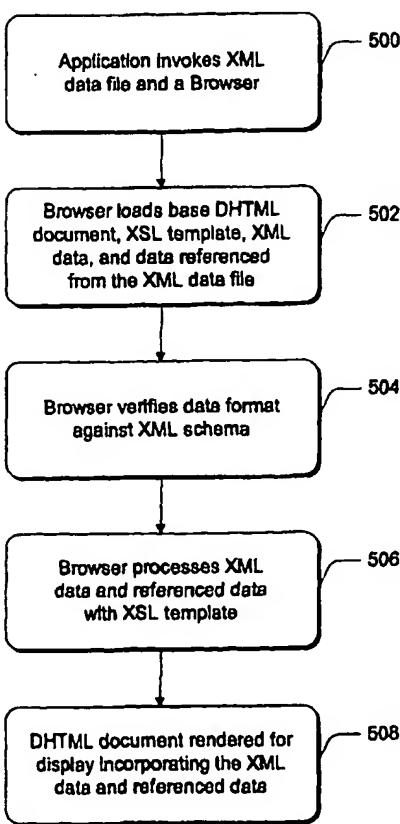
(74) Agents: **MÖRASCH, David, A. et al.**; Suite 500, 421 W. Riverside Avenue, Spokane, WA 99201 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SI, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: METHODS AND SYSTEMS FOR DYNAMICALLY CREATING USER INTERFACES



(57) Abstract: A data presentation system dynamically generates and renders a user interface (UI) in the form of a Hypertext Markup Language (HTML) document. Extensible Markup Language (XML) data files contain data that defines the UI. The XML data is maintained independently of any application and/or computing system's settings and controls that define the presentation of the UI. The XML data files also have reference links to secondary data files that contain data further defining the UI. The UI is dynamically generated when an application invokes the XML data files. The HTML document incorporates the data from both the XML data files and from the secondary data files when rendering the UI.

WO 01/44932 A1



patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:

— *With international search report.*

Methods and Systems for Dynamically Creating User Interfaces**RELATED APPLICATION**

5 This application claims priority from U.S. Provisional Application Serial No. 60/170958, filed December 15, 1999, the disclosure of which is incorporated herein by reference.

TECHNICAL FIELD

10 This invention relates to the systems and methods for implementing an extensible, dynamically created user interface. Specifically, the presentation of a control panel user interface for a computing system's settings and controls is dynamically created using Web-based technology.

15 **BACKGROUND OF THE INVENTION**

Access to a computing system's settings and controls are typically presented to a user via a collection of individual user interactive applications collected for presentation within a control panel user interface. The computing system's settings and controls are coded with a programmable means for user interaction at the user 20 interface.

Figure 1 shows a computing system 100 having a conventional control panel user interface. Computing device 100 has a display 110 and a base 112. The base 112 contains memory 114 that stores systems applications 116 and 118(A₁-A_N).

A user interface (UI) 120 is shown displayed on the computing device 25 display 110. In Figure 1, the UI 120 is embodied as the control panel 130 in the familiar "Windows"-based operating system from Microsoft Corporation. Control

panel 130 presents a user with a multitude of selectable application control icons 132(I₁-I_N) that expose both simple and complex system configuration settings and controls at the same level. Each application icon 132 is independently associated with a system application 118 within the computing device memory 114 (e.g., icon 5 I₁ is the independent UI associated with application A₁, and so forth).

The presentation of control panel 130 is controlled by system application 116. Each systems application 118 is associated with the presentation of a control panel UI, and is also integrated with the presentation of the control panel 130 in conjunction with the control panel system application 116.

10 From a user standpoint, multiple unrelated system control panels make it difficult for a user to locate a system setting or control that he or she wants to change. Furthermore, related settings and controls are separated into different control panels, causing a user to have to search the various control panels to accomplish a set of related settings and control tasks.

15 Accordingly, there is a need for control panel UIs that enables a user to easily navigate a computing system's settings and controls. Additionally, there is a need for an extensible UI presentation technique that enables a developer to easily design, modify, and replace control panel UIs.

20 New computing display and data presentation techniques have developed as a result of the Internet and the fast-paced technological advancements in computing and user interaction. One such presentation and display technology utilized by developers of Internet Web pages is Hypertext Markup Language (HTML) in conjunction with Extensible Markup Language (XML). Many readers will be familiar with HTML and XML. However, to provide additional background, the 25 following discussion presents an overview of the conventional HTML and XML technologies.

Hypertext Markup Language

Hypertext Markup Language (HTML) is a display language for presenting data to an end user in a visual presentation. Internet Web pages are commonly written in HTML and may include text, images, sound, video, active code, and so forth. Data binding is an aspect of Dynamic HTML (DHTML) that allows processing individual items of data from an information source into an HTML display, thus allowing HTML to be used as a template for displaying the data, while creating an extensible and interactive Web page. DHTML allows a developer to individually manipulate any HTML page element at any time.

Making simple updates to an HTML page, such as changing the color of text after a Web page has been rendered for display, typically requires reloading the entire page to account for the one desired change. With DHTML, however, every element within the page is independently dynamic and an individual element (text or graphics) can be changed without having to reload the entire page.

Extensible Markup Language

Extensible Markup Language (XML) is a meta-markup language that provides a format for describing structured data. XML is similar to HTML in that it is a tag-based language. By virtue of its tag-based nature, XML defines a strict tree structure or hierarchy. XML is a derivative of Standard Generalized Markup Language (SGML) that provides a uniform method for describing and exchanging structured data in an open, text-based format. XML utilizes the concepts of elements and namespaces. Compared to HTML, which is a display-oriented markup language, XML is a general purpose language for representing structured data without including information that describes how to format the data for display.

Internet Web-based data viewing is typically enabled via XML data that is stored on a server at an Internet site. The XML data is transferred to a client device upon request and rendered for viewing with HTML. An XML document does not by itself specify whether or how its data should be displayed. The XML data is 5 merely the factual information to be displayed.

An unique attribute of utilizing XML structured data is that the data is separated from the HTML presentation to a user. While HTML specifies how to display data in an Internet browser, XML only defines the content data to be displayed. HTML utilizes tags to specify how an Internet browser will display the 10 data (i.e., bold or italic). In XML, tags are only used to describe the data (i.e., a title name), and not to specify how the data will be displayed. To describe how the data will be presented, XML references a separate application or stylesheet.

Extensible Stylesheet Language

Extensible Stylesheet Language (XSL) provides both the semantic and 15 structural independence for presenting the content of XML data. A style sheet contains a template of the desired structure for data presentation, and identifies the data in an XML source document to insert into the template. Thus, the XML data is separated from the presentation, allowing both the grammar and the structure of the 20 XML source document to be independent of the presentation language and the structure.

XSL style sheets contain instructions for presenting the data in an XML document in a format such as HTML. XML enables data to be processed and displayed by applying different stylesheets and applications while separating the data from the presentation and the process. The separation of the data from the 25 presentation allows for a seamless integration of data from diverse sources. This

further allows data encoded in XML to be easily exchanged on-line and delivered over the Internet to a client device.

Figure 2 shows the components for transforming XML-based data into an HTML display using XSL. The HTML presentation system 200 includes an 5 Internet server 202 and a requesting client device 204. The Internet server 202 includes XML data 210, an XSL template 212, and an XSL processor 214. The XML data 210 is processed with the XSL template file 212 at the XSL processor 214 to generate and present an HTML document 216 at the client device 204.

XML Structure

10 XML “elements” are structural constructs that consist of a start tag, an end or close tag, and the information or content that is contained between the tags. A “start tag” is formatted as “<tagname>” and an “end tag” is formatted as “</tagname>”. In an XML document, start and end tags can be nested within other 15 start and end tags. All elements that occur within a particular element must have their start and end tags occur before the end tag of that particular element. This defines a strict tree-like structure. Each element forms a node in this tree, and potentially has “child” or “branch” nodes. The child nodes represent any XML elements that occur between the start and end tags of the “parent” node.

Namespaces in XML structured data qualify element names in a 20 recognizable manner to avoid conflicts between elements with the same name. Namespaces ensure that element names do not conflict, and they clarify who defined which term. They do not give instructions on how to process the elements, but simply keep the names straight. Thus, XML namespaces convey meaningful information by associating a descriptive namespace with a tag name.

25 Within an XML document, namespace declarations occur as attributes of start tags. Namespace declarations are of the form “xmlns:[prefix]=[uri]”. A

namespace declaration indicates that the XML document contains element names that are defined within a specified namespace or schema. A “prefix” is an arbitrary designation that will be used later in the XML document as an indication that an element name is a member of the namespace declared by a universal resource indicator (uri). The prefix is valid only within the context of the specific XML document. A uri is either a path to a document describing a specific namespace or schema, or a globally unique identifier of a specific namespace or schema. A uri is valid across all XML documents. Namespace declarations are “inherited”, which means that a namespace declaration applies to the element in which it was declared as well as to all elements contained within that element.

Namespace inheritance within an XML document allows non-qualified names to use “default” namespaces. Default namespaces are explicitly declared as attributes of start tags. Default namespace declarations are of the form “xmlns=[uri]”. Note that the declaration of a default namespace is equivalent to the declaration of a non-default namespace but the prefix is omitted. A namespace specification within an XML document is said to have a “scope” which includes all child nodes beneath the namespace specification.

An example of XML code follows:

```
<trans:orders xmlns:person="http://www.schemas.organization/people"
               xmlns:dsig="http://www.dsig.org"
               xmlns:trans="http://www.schemas.org/transactions">
  <trans:order>
    <trans:sold-to>
      <person:name>
        <person:last-name>Layman</person:last-name>
        <person:first-name>Andrew</person:first-name>
      </person:name>
```

```
</trans:sold-to>
<trans:sold-on>1997-03-17</trans:sold-on>
<dsig:digital-signature>1234567890</dsig:digital-signature>
</trans:order>
</trans:orders>
```

This code includes three XML namespace declarations that are each designated with "xmlns". The declarations include a prefix (e.g. "person", "trans", and "dsig") and the expanded namespace to which each prefix refers (e.g. 5 "http://www.schemas.org/people", "http://www.schemas.org/transactions", and "http://www.dsig.org", respectively). This code tells any reader that if an element name begins with "dsig" its meaning is defined by the "http://www.dsig.org" namespace. Similarly, elements beginning with the "person" prefix have meanings defined by the "http://www.schemas.org/people" namespace and elements 10 beginning with the "trans" prefix have meanings defined by the "http://www.schemas.org/transactions" namespace.

It is important to note that another XML document that incorporated elements from any of the namespaces included in this sample might declare prefixes that are different from those used in this example. As noted earlier, prefixes are 15 arbitrarily defined by the document's author and have meaning only within the context of the specific element of the specific document in which they are declared. Tags from multiple namespaces can be mixed to accommodate data coming from multiple sources. With namespaces, two elements can exist in the same XML-based document, but can refer back to two different schemas, uniquely qualifying 20 their semantics.

XML Schemas

XML accommodates an infinite number of database schemas and elements referenced in one XML document can be defined in different schemas. XML

schema are typically implemented within a Web browser such as Internet Explorer from Microsoft Corporation to provide support for coinciding sub-sets of XML data. An XML schema specifies the structure of an XML document and the constraints on the document's content.

5 Specifically, a schema defines the elements that can appear within an XML document and the attributes that can be associated with an element. While defining the structure of an XML document, a schema also defines which elements are child elements of others, the sequence in which the child elements can appear, and the number of child elements. A schema further defines whether an element is empty 10 or can include text, as well as defining default values for attributes. Overall, a schema is utilized to validate the content of an XML document and to determine whether an XML document is a valid instance of the grammar expressed by the schema.

Within each schema, a "dictionary" of element names is defined. The 15 dictionary of element names defined by a schema is referred to as a "namespace." Within an XML document, element names are qualified by namespace identifiers. When qualified by a namespace identifier, a tag name appears in the form "namespace:[tagname]". This model enables the same element name to appear in multiple schemas, or namespaces, and for instances of these duplicate element 20 names to appear in the same XML document without colliding. Start tags can declare an arbitrary number of "attributes" which declare "property values" associated with the element being declared. Attributes are declared within the start tag using the form "<[tagname] [attribute1],[attribute2]...,[attributeN]>", where attribute1 through attributeN are declarations of an arbitrary number of tag 25 attributes. Each attribute declaration is of the form

“[attributeName]=[attributeValue]” where each attribute is identified by a unique name followed by an “=” character, followed by the value of the attribute.

A schema is primarily the formal specification of the elements and attributes allowed within a particular class of XML document, and the schema specifies how 5 those elements and attributes are related to each other. Elements and attributes are defined in an XML schema document by specifying an <ElementType...> and <AttributeType...>, respectively. These provide the definition and type of the element or attribute. An instance of an element or an attribute is declared using <element...> or <attribute...> tags. An example of an XML schema is shown 10 following:

```
<?xml version="1.0"?>
<Schema xmlns="schemas-microsoft-com:XML-data">
    <ElementType name="title" />
    <ElementType name="author" />
    <ElementType name="pages" />
    <ElementType name="book" model="closed">
        <element type="title" />
        <element type="author" />
        <element type="pages" />
        <AttributeType name="copyright" />
        <attribute type="copyright" />
    </ElementType>
</Schema>
```

In this example, there are four <ElementType> defining elements: "title," "author," "pages," and "book." Within the ElementType for "book," the content 15 model for a book can be declared. Each book contains "title," "author," and "pages" elements using the <element> with a type attribute that references the ElementType. An <AttributeType> for the copyright attribute can also be defined and then its usage declared using the <attribute> element with a type attribute that references its definition.

The definition of the copyright attribute is contained within the ElementType for "book." Attribute definitions are distinct from ElementType definitions in that they can be declared within the scope of an ElementType, allowing different element types to declare attributes of the same name but with potentially different meaning. 5 <AttributeType> elements can be declared globally by placing them outside of the context of an ElementType. In this way, multiple elements can share the definition of a common attribute without having to redeclare the AttributeType within each ElementType.

10 SUMMARY OF THE INVENTION

A data presentation system utilizes independent data file(s) as a way to define user interfaces (UIs) that are extensible, easy to redesign, and to update. The data presentation system separates a computing system's settings applications from the user interface (UI) used to set the system settings. This allows a UI to be 15 dynamically modified simply by changing the description of the UI in the independent data file(s).

The system architecture includes a namespace navigation architecture that allows developers to access a computing system's settings and controls without having to know the details of the corresponding system files. System files' details 20 include the system files' physical location, file permissions, and file names. Utilizing a namespace navigation architecture, one can simply refer to a system setting or control and navigate directly to it. This is essentially creating a URL (universal resource locator) to navigate and organize system resources.

In one implementation, the data presentation system dynamically generates a 25 Hypertext Markup Language (HTML) user interface. The HTML user interface is generated with independent Extensible Markup Language (XML) data files

containing data that defines the UI. The XML data files also contain namespace reference links to secondary and independent data files containing data that further defines the UI. The data presentation system renders the HTML user interface for display, incorporating the data from both the XML data files and from the 5 secondary data files, by invoking the XML data files only.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the drawings to reference like features and components.

10 Fig. 1 is a block diagram that illustrates a computing device displaying a user interface.

Fig. 2 is a block diagram that illustrates an Internet server providing data for presentation on an Internet browser at a client device.

Fig. 3 is a block diagram of an exemplary computing device.

15 Figs. 4A, 4B, 4C, and 4D are block diagrams illustrating an exemplary data presentation system that is implemented by the computing device shown in Figure 3.

Fig. 5 is a flow diagram that describes steps in an exemplary data presentation method.

20 Fig. 6 is a diagram illustrating the structure of a control panel hierarchy.

Fig. 7A illustrates an exemplary control panel user interface.

Fig. 7B illustrates a control panel user interface accessible via the control panel user interface illustrated in Figure 7A.

Fig. 8A illustrates an exemplary control panel user interface.

25 Fig. 8B illustrates a control panel user interface accessible via the control panel user interface illustrated in Figure 8A.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The data presentation system and methods described herein separates a displayable user interface from a computing system's applications and the system's settings and controls that define the presentation of the user interface. By utilizing independent data files, a user interface (UI) can be dynamically modified simply by changing the description of the UI in the independent data files. Thus, the UI is extensible, easy to redesign, update, or extend.

The data presentation system utilizes Internet Web-based technology on an independent computing device to dynamically generate a displayable document. Dynamically displaying a Hypertext Markup Language (HTML) document generated with Extensible Markup Language (XML) data facilitates incorporating secondary and independent document defining data into the document when displayed.

15 Exemplary Computing Device

Figure 3 illustrates an example of an independent computing device 300 that can be used to implement the components of a data presentation system. Computing device 300 includes one or more processors or processing units 302, a system memory 304, and a bus 306 that couples the various system components including the system memory 304 to processors 302. The bus 306 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory 304 includes read only memory (ROM) 308 and random access memory (RAM) 310. A basic input/output system (BIOS) 312, containing the basic routines that help to transfer

information between elements within the computing device 300 is stored in ROM 308.

Computing device 300 further includes a hard drive 314 for reading from and writing to one or more hard disks (not shown). Some computing devices can 5 include a magnetic disk drive 316 for reading from and writing to a removable magnetic disk 318, and an optical disk drive 320 for reading from or writing to a removable optical disk 322 such as a CD ROM or other optical media. The hard drive 314, magnetic disk drive 316, and optical disk drive 320 are connected to the bus 306 by a hard disk drive interface 324, a magnetic disk drive interface 326, and 10 a optical drive interface 328, respectively. Alternatively, the hard drive 314, magnetic disk drive 316, and optical disk drive 320 can be connected to the bus 306 by a SCSI interface (not shown).

The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and 15 other data for computing device 300. Although the exemplary environment described herein employs a hard disk 314, a removable magnetic disk 318, and a removable optical disk 322, it should be appreciated by those skilled in the art that other types of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, 20 random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on ROM 308, RAM 310, the hard disk 314, magnetic disk 318, or optical disk 322, including an operating system 330, one or more application programs 332, other program modules 334, 25 and program data 336. In some computing devices 300, a user might enter commands and information into the computing device 300 through input devices

such as a keyboard 338 and a pointing device 340. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. In some instances, however, a computing device might not have these types of input devices. These and other input devices are connected to the processing unit 302 through an interface 342 that is coupled to the bus 306. In some computing devices 300, a monitor 344 or other type of display device might also be connected to the bus 306 via an interface, such as a video adapter 346. Some devices, however, do not have these types of display devices. In addition to the monitor 344, computing devices 300 might include other peripheral output devices (not shown) such as speakers and printers.

Generally, the data processors of computing device 300 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computing device 300. At execution, they are loaded at least partially into the computing device's primary electronic memory. The computing devices described herein include these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the steps described below in conjunction with a microprocessor or other data processor. The service system also includes the computing device itself when programmed according to the methods and techniques described below.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times

in different storage components of the computing device 300, and are executed by the data processor(s) of the computer.

Data Presentation System

Figures 4A and 4B show the components of a data presentation system 400 implemented at the computing device 300 of Figure 3. Computing device 300 has a processor 302, a system memory 304 (or other memory such as a hard disk, a removable magnetic disk, and/or a removable optical disk as shown in Fig. 3), and a display 344. The memory 304 retains XML schema 410, an XSL template 412, XML data 414, and a mouse control panel file 416 located in memory 304 at a file space location c:\winnt\system32\main.cpl. The mouse control panel file 416 contains mouse control panel UI settings and controls.

The processor 302 is shown executing an application 418 and a browser 420. In response to the execution of application 418, browser 420 loads the XML data 414 and checks the data file against XML schema 410 to ensure that the XML data 414 is a valid representation of the schema's specified content. The browser 420 then processes the XML data 414 with the XSL template 412 at an XSL processor 422 located within browser 420 to render a DHTML document 424 on the display 344. An example of an XSL processor 422 is a dynamically linked library named MSXML.DLL available in the Internet Explorer browser from Microsoft Corporation. A dynamically linked library (DLL) is a binary file that acts as a shared library of functions that can be used simultaneously by multiple applications. DLLs are used for a variety of reasons, primarily for either sharing common code between different executable files or breaking an application into separate components, thus allowing for easy upgrades of the application.

In Figure 4A, application 418 invokes XML data 414 that contains a namespace navigation link 426 to the mouse control panel file 416 located in

memory 304 at c:\winnt\system32\main.cpl. The browser 420 processes the XML data 414 with the XSL template 412 at the XSL processor 422 to render the DHTML document 424 that forms a mouse control panel UI on the display 344. The DHTML document 424 forming the mouse control panel UI incorporates the 5 mouse control panel settings and controls located in mouse control panel file 416.

Figure 4A demonstrates that a DHTML document 424 is created for display and incorporates data from the file 416 by simply invoking the XML data 414 that contains the namespace navigation link 426 to the file 416. Namespace navigation link 426, /controlpanel/hardware/mouse, is a reference link of navigable nodes that 10 is part of a namespace navigation architecture that describes a system's settings and controls using simple, descriptive, and easy to understand terminology. It is not clear from the file path, c:\winnt\system32\main.cpl, that this points to a mouse control panel file 416. The namespace navigation link 426, however, provides a user with an intuitive understanding that a mouse control panel file can be accessed 15 via the namespace path, /controlpanel/hardware/mouse.

A particular advantage of utilizing a namespace navigation architecture is that it provides a consistent and understandable interface means for application developers and independent software vendors (ISVs) to use and invoke a particular system's settings and controls. In the illustrated example of Fig. 4A, an application 20 developer desiring to display a user interface with mouse settings and controls would not have to know the specific file location for the mouse control panel file 416 in memory 304 to include the settings and controls in a mouse control panel user interface (e.g., DHTML document 424).

The namespace navigation architecture allows developers and ISVs to access 25 a computing system's settings and controls without having to know the details of the corresponding system files. System files' details include the system files'

physical location, file permissions, and even file names. By referencing the namespace navigation architecture (i.e., the reference link 426, /controlpanel/hardware/mouse), one can refer to a data file (i.e., mouse control panel file 416) containing user interface settings and controls and navigate directly 5 to it without having to know the file specific details. This is essentially creating a URL (universal resource locator) to navigate system resources.

In Figure 4B, a new mouse control panel file 428 supersedes the previous mouse control panel file 416 and is located in memory 304 at a file space location c:\winnt\system32\mouse.cpl. The new file 428 contains new mouse control panel 10 UI settings and controls that application 418 incorporates into the mouse control panel UI, DHTML document 424.

If application 418 had originally directly referenced the previous file 416, application 418 would have to be reprogrammed to invoke the new mouse control panel file 428 at the new location in memory 304. The ramifications of relocating 15 data from an existing file to a new file at a new location in memory during data and application development, and during upgrades, are compounded for each application and system component referencing an existing file.

In contrast, according to the data presentation system described herein, the only change needed from a development standpoint is a simple text edit in XML 20 data 414 so that the namespace hierarchy link 426 now references the new mouse control panel file 428. Developing the new mouse control panel file 428 and relocating the file to a new location in memory 304 does not require any reprogramming of application 418, or of any other proprietary or third party applications invoking the XML data 414.

25 In the implementation, the XML data 414 is configured and stored in a manner that describes a namespace according to the specified content in the XML

schema 410. All nodes in a namespace have a single parent node, except for the root node which does not have a parent node. Thus, the hierarchy of a namespace reference link is described through the physical structure of the XML data document itself. The purpose of the namespace navigation architecture illustrated 5 in Figures 4A and 4B is to derive the contents of a displayable user interface with a simple navigation hierarchy.

One advantage of the described architecture is that XML schemas and XML data files are extensible and can be easily modified. A developer can add elements and attributes to a schema document and/or data file to support a particular user 10 interface presentation. Utilizing XML, XSL, and DHTML allows a displayable UI to appear more like a Web page and further allows the incorporation of Web-like controls with which Internet users are already intuitively familiar. In addition, the UI can be rendered with dynamically generated help text, tooltips, and the like.

Figures 4C and 4D illustrate the extensibility of utilizing a namespace 15 navigation architecture to invoke independent XML data files to define a UI. In Figure 4C, Application 430 executes on processor 302 and has a namespace reference link 432 to XML data file 434 located in memory 304 at a file space location c:\winnt\system32\main.cpl. The XML data file 434 contains mouse control panel UI settings and controls.

20 When the application 430 is executed on processor 302, browser 420 processes the XML data 434 with the XSL template 412 at the XSL processor 422 to render a DHTML mouse control panel UI document 424 on display 344. The DHTML document 424 forming the mouse control panel UI incorporates the mouse control panel settings and controls located in the XML data file 434.

25 Figure 4D illustrates that the XML data file 434 has been replaced with a new XML data file 436 which is located in memory 304 at location

c:\winnt\system32\mouse.cpl. This new memory location differs from the file space location of the now superseded XML data file 434.

When the application 430 is again executed on processor 302, the browser 420 processes the XML data 436 with the XSL template 412 at the XSL processor 5 422 to render a new DHTML mouse control panel UI document 438 on display 344.

The example illustrated in Figs. 4C and 4D shows that a designer or systems programmer can dynamically modify a UI simply by changing the description of the UI in the XML data file that defines the UI. The example also shows that the abstraction of a system's settings and controls into a namespace navigation 10 architecture eliminates the need for application and systems programmers to have to hard-code file space locations into software applications. Specifically, application 430 is able to call a data file containing the mouse control panel settings and controls with the namespace reference link 432 even after the XML data is modified and after the data file is moved to a new location in memory 304 (Fig. 4C 15 to Fig. 4D).

Figure 5 shows a method for generating and rendering data with the data presentation system shown in Figures 4A-4B. The method is implemented, and the steps are performed, in software. At step 500, an application executing on a computing device processor invokes an XML data file and a browser. The XML 20 data file references and incorporates other independent data file(s).

In response to executing the application, the browser loads a base DHTML document, an XSL template, the XML data, and the referenced data at step 502. The base DHTML document initiates the browser to load the XSL template and the XML data file and then generates a DHTML document from the XML data using 25 the XSL template. At step 504, the browser verifies that the data format is a valid representation of an XML schema's specified content. At step 506, after verifying

the data format, the browser processes the XML data and the referenced data with the XSL template. A resultant DHTML document is rendered at a computing device display at step 508. The resultant DHTML document incorporates the XML data invoked by the application at the initial step 500 and incorporates the data 5 referenced by the XML data file.

Control Panel Hierarchy

Figure 6 illustrates an overall structure of a control panel hierarchy 600. The control panel pages are DHTML documents rendered with the data presentation system shown in Figures 4A-4B. Navigation pages 602 and 604 have prompts, 10 integrated help features, and selectable namespace navigation links to groups of system settings and controls at task pages 606 and 608. Task pages 606 and 608 have selectable namespace navigation links to facilitate a user when changing aspects of a computing device's system settings and controls, to include system operation and display presentation.

15 A home page 610 is also a navigation page from which a user can select a namespace navigational link to a task page 612. A home page 610 is a user starting point for a group of closely related system's settings and/or controls. A home page generally contains only namespace navigation links to individual settings or controls on a task page. Furthermore, a home page also includes status content to 20 inform a user of the system status, a current setting, or other content associated with the particular system setting and/or control on a particular home page.

By organizing related settings and controls together on control panel pages 602 through 612 in the control panel hierarchy 600 according to usability factors, and using control-orientated terminology, a user intuitively understands the 25 system's settings and the associated controls. The organization of the control panel hierarchy 600 allows for cross-linking and multiple control panel page access to

system settings and the associated controls wherever a user might look for a particular setting and/or control. Because each setting and control is written as an individual ActiveX control embedded in a DHTML page, the same setting and/or control can easily be made available at multiple logical control panel pages.

5 Help features are integrated at the control panel pages to present help information that is contextually related to the particular control panel page currently rendered for display. A search for a particular help topic generates selectable links that direct a user to the appropriate setting task on a particular control panel page.

Control Panel User Interface

10 Figures 7A and 7B illustrate examples of a DHTML document generated by the data presentation system 400 shown in Figures 4A-4D. Control panel navigation pages 602 and 604 are shown having selectable namespace navigation links 700 and 702. Control Panel navigation page 604 (Fig.7B) also shows selectable namespace sub-navigation links 704. Each selectable navigation link 15 700, 702, and 704 is a node link in a namespace that is generated dynamically. A namespace is a hierarchy of nodes similar to that of a URL (universal resource locator) and each node in a namespace is similar to an element in the URL path. A node is identified by its "ID" which is simply the name of the displayed selectable navigation link on the control panel pages 602 and 604.

20 The root node of a namespace navigation link is the ID of the navigation link. In Figure 7A, the root node of the namespace for control panel 602 is ID="ControlPanel". Any node in the namespace architecture (e.g., selectable links 700, 702, and 704) is uniquely identified by its full path name from the root node. Node link 706 in Figure 7A is identified by the path name 25 "/ControlPanel/Settings/AppearanceThemes". Node link 702 in Figure 7B is identified by the path name "/ControlPanel/Settings/AppearanceThemes/Themes".

Figure 7A shows that specific node link 706, which identifies namespace node "/ControlPanel/Settings/AppearanceThemes", navigates through the namespace architecture to render Control Panel page 604 (Fig. 7B) at a computing device display as indicated by arrow 708. Figure 7A also shows that node link 710, 5 which identifies node "/ControlPanel/Settings/AccountsParentalControls", navigates through the namespace architecture to initiate rendering Control Panel page 610 (Fig. 8A) at a computing device display as indicated by arrow 712.

Figures 8A and 8B illustrate examples of a DHTML document generated by the data presentation system 400 shown in Figures 4A-4D. Figure 8A illustrates 10 control panel home page 610 in the control panel hierarchy 600. The home page 610 is shown having two individual setting namespace node links 800 and 802 to task pages. Home page 610 also shows status content 804 to inform a user of the current account settings. Selecting node link 802 (i.e., "Change name") navigates 15 through the namespace architecture to render control panel task page 612 (Fig. 8B) at a computing device display as indicated by arrow 806.

Figure 8B illustrates a control panel task page 612. The user interface has a primary area 810 (the right-side pane) and a secondary area 812 (the left-side pane). The primary area 810 is a DHTML page dynamically created using source HTML or independent XML data and hosting ActiveX controls, some of which are 20 exposed through "Windows"-brand Management Instrumentation (WMI). The presentation system 400 hosts the DHTML page and uses the source HTML or independent XML data to construct the secondary area 812. The secondary area 812 displays help content 814 that is selectable by a user. The primary area 810 is presented to assist a user in completing a particular task. The primary area 810 25 displays a dynamically generated task 816, "Type a new account name", that is a link within the namespace navigation architecture. The primary area 810 also

presents the user with confirmation buttons 818, “Change Name” and “Cancel”, to either enact or cancel a setting change.

Some “Windows”-brand control panels that exist in the Win32 user interface format are more beneficial to a user in the current format than they would be if 5 presented in the DHTML format. In such a case, a Win32 control panel can be incorporated into the control panel hierarchy 600 (Fig. 6) for access by a user. In addition, third party control panel user interfaces can be incorporated into the control panel hierarchy 600.

Control Panel XML Schema Document

10 The following XML schema (i.e., item 410 in Fig. 4A-4D) defines the namespace for Control Panel and specifies the elements and attributes allowed for the control panel pages illustrated in Figures 7A-7B and 8A-8B in the control panel hierarchy 600 shown in Figure 6. Each Control Panel namespace node in the schema 410 is represented by a CPNS node (control panel namespace node) 15 element in an XML data document (i.e., item 414 in Figs. 4A-4D).

```
<?xml version="1.0"?>
<Schema name="ms-ControlPanel" xmlns="urn:schemas-microsoft-com:xml-data"
       xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <!-- ID is the name of this node, no two siblings can have the same ID -->
    <AttributeType name="ID" dt:type="nmtoken" required="yes"/>

    <!-- access permissions -->
    <AttributeType name="PERMISSIONS" dt:type="enumeration"
                   dt:values="owner child guest all"/>

    <!-- extensible implies that this node can have children nodes -->
    <AttributeType name="EXTENSIBLE" dt:type="enumeration"
                   dt:values="yes no"/>

    <!-- replaceable implies that this node can be replaced by another node -->
    <AttributeType name="REPLACEABLE" dt:type="enumeration"
                   dt:values="yes no"/>

    <!-- how many children of this node will be exposed in lists -->
    <AttributeType name="NUMEXPOSED" dt:type="int" dt:min="0"/>
```

```

<!-- priority amongst sibling nodes, 0 is lowest priority, 1 is highest -->
<AttributeType      name="PRIORITY" dt:type="float" dt:min="0.0"
                      dt:max="1.0"/>

<!-- Extensions to the ControlPanel namespace can be made by specifying
the MOUNTPATH attribute on a root CPNSNode in an XML file (A root
CPNSNode is one which does not have another CPNSNode as its parent).
The root CPNSNode is then attached as the child node of the node
specified by the MOUNTPATH. MOUNTPATH attributes on a non-root
CPNSNode are ignored. -->

<AttributeType      name="MOUNTPATH"      dt:type="string"
                      required="yes"/>

<!-- The following tag names are in sync with the command presentation
schema tags. This is for convenience and to reflect the fact that these tags
are used as is while generating the command presentation XML file. -->

<ElementType name="Title"           content="textOnly"/>
<ElementType name="ShortTitle"      content="textOnly"/>
<ElementType name="Description"     content="textOnly"/>
<ElementType name="ToolTip"         content="textOnly"/>
<ElementType name="Image"           content="textOnly"/>
<ElementType name="Target"          content="textOnly"/>
<ElementType name="SearchTarget"    content="textOnly"/>
<ElementType name="SearchKey"       content="textOnly"/>

<!-- Control Panel specific tags -->

<ElementType name="Directive"      content="textOnly"/>
<ElementType name="SeeAlso"         content="textOnly"/>
<ElementType name="TellMeMore"      content="textOnly"/>
<ElementType name="ContentHTML"     content="textOnly"/>

<!-- Control Panel Namespace node -->

<ElementType name="CPNSNode">
  <attribute type="ID"/>
  <attribute type="PERMISSIONS"      default="all"/>
  <attribute type="EXTENSIBLE"       default="no"/>
  <attribute type="REPLACEABLE"      default="no"/>

  <element type="Title"             minOccurs="0" maxOccurs="1"/>
  <element type="ShortTitle"        minOccurs="0" maxOccurs="1"/>
  <element type="Description"       minOccurs="0" maxOccurs="1"/>
  <element type="Tooltip"           minOccurs="0" maxOccurs="1"/>
  <element type="Image"             minOccurs="0" maxOccurs="1"/>
  <element type="Target"            minOccurs="0" maxOccurs="1"/>
  <element type="SearchTarget"      minOccurs="0" maxOccurs="1"/>
  <element type="Directive"         minOccurs="0" maxOccurs="1"/>
  <element type="ContentHTML"       minOccurs="0" maxOccurs="1"/>
  <element type="CommandManifest"    minOccurs="0" maxOccurs="1"/>
  <element type="SearchKey"         minOccurs="0" maxOccurs="*"/>

```

25

```
<element type="SeeAlso" minOccurs="0" maxOccurs="*"/>
<element type="TellMeMore" minOccurs="0" maxOccurs="*"/>

<!-- Recursively include nodes -->
<element type="CPNSNode" minOccurs="0" maxOccurs="*"/>
</ElementType>
<!-- CPNSNode -->
</Schema>
```

The attribute and element types specified in the XML schema document above are described following:

AttributeType “MOUNTPATH” defines an attribute on a root CPNSNode 5 (control panel namespace node) in an XML file that can be utilized to specify an extension to the ControlPanel namespace.

Attribute type “ID” defines a control panel namespace identifier.

Attribute type “PERMISSIONS” defines access permission for the setting. The default value is (ALL), but some categories of users may be denied from 10 changing certain system settings.

Attribute type “EXTENSIBLE” designates that the control panel namespace node will not have any children nodes. The default value is (NO) to define whether or not a 3rd party can add children to a particular node/branch of the taxonomy.

Attribute type “REPLACEABLE” designates that the control panel 15 namespace node cannot be replaced by another node. The default value is (NO) to define whether or not a particular node/branch is completely replaceable in the future.

ElementType “Title” defines the name of the node, page, or branch that appears in the UI.

20 ElementType “ShortTitle” defines a shorter version of “Title”.

ElementType “Description” defines, in plain text, the function of a node.

ElementType “ToolTip” defines the ToolTip text for a node link to this page.

ElementType “Image” defines an image to be displayed on a page.

ElementType “Target” defines the navigation target or the command target for this node.

ElementType “SearchTarget” defines the page to navigate to for a search.

5 ElementType “SearchKey” defines keywords that describe the content of this page that are used in a search.

ElementType “Directive” defines UI text that tells the user what to do on this page.

10 ElementType “SeeAlso” defines links to related nodes and branches of the taxonomy.

ElementType “TellMeMore” defines links to help topics that are related to this node or branch.

ElementType “ContentHTML” defines the location of the actual HTML file which appears in the content pane.

15 ElementType “CPNSNode” defines the data as a control panel namespace node.

Element type “CommandManifest” defines the location of the file which defines the commands for this page.

20 Although the data presentation system has been described in language specific to structural features and/or methodological steps, it is to be understood that the systems, architectures, and methods defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed 25 methods, architectures, and systems.

CLAIMS

We claim:

1. A control panel user interface (UI) presentation system comprising:
 - 5 an extensible markup language (XML) data file having XML data configured to define a control panel UI;
 - a second data file having secondary data configured to further define the control panel UI;
 - the XML data file having a reference link to the second data file; and
 - 10 a browser to process the XML data and the second data to dynamically generate the control panel UI.
2. A control panel user interface (UI) presentation system as recited in claim 1, further comprising an XML schema configured to define a format of the control panel UI data in the XML data file and in the second data file.
 - 15
3. A control panel user interface (UI) presentation system as recited in claim 1, further comprising an extensible stylesheet language (XSL) template wherein the browser has an XSL processor to process the XML data and the second data to dynamically generate the control panel UI.
 - 20
4. A control panel user interface (UI) presentation system as recited in claim 1, wherein the control panel UI is a dynamically generated hypertext markup language (HTML) document.

5. A control panel user interface (UI) presentation system as recited in claim 1, wherein the browser renders the control panel UI, the control panel UI incorporating the control panel UI data contained in the XML data file and in the second data file.

5

6. A control panel user interface (UI) presentation system as recited in claim 1, further comprising a display to render the control panel UI, the control panel UI incorporating for display the control panel UI data contained in the XML data file and in the second data file.

10

7. A control panel user interface (UI) presentation system as recited in claim 1, wherein the reference link is a subset of navigable nodes in a navigation architecture, the reference link being configured to intuitively describe the second data.

15

8. A control panel user interface (UI) presentation system comprising:
an extensible markup language (XML) data file having data configured to define a control panel UI;
one or more secondary data files having control panel UI data configured to further define the control panel UI;
the XML data file having a reference link to the one or more secondary data files;
an XML schema configured to define a format of the control panel UI data in the XML data file and in the one or more secondary data files;
20 an extensible stylesheet language (XSL) template; and
25

a browser having an XSL processor to process the control panel UI data with the XSL template to dynamically generate the control panel UI.

9. A control panel user interface (UI) presentation system as recited in
5 claim 8, wherein the control panel UI is a dynamically generated hypertext markup
language (HTML) document.

10. A control panel user interface (UI) presentation system as recited in
claim 8, wherein the browser renders the control panel UI, the control panel UI
10 incorporating the control panel UI data contained in the XML data file and in the
one or more secondary data files.

11. A control panel user interface (UI) presentation system as recited in
claim 8, further comprising a display to render the control panel UI, the control
15 panel UI incorporating for display the control panel UI data contained in the XML
data file and in the one or more secondary data files.

12. A control panel user interface (UI) presentation system as recited in
claim 8, wherein the reference link is a subset of navigable nodes in a navigation
20 architecture, the reference link being configured to intuitively describe the control
panel UI data in the one or more secondary data files.

13. A data presentation system comprising:
a first data file having first data configured to define a document;
25 a second data file having second data to further define the document;
the first data file having a reference link to the second data file; and

an application configured to invoke the first data file to generate the document, wherein invoking the first data file initiates the reference link to invoke the second data file so that the document contains the first and second data when generated.

5

14. A data presentation system as recited in claim 13, further comprising a display to render the document, the document incorporating for display the first and second data.

10

15. A data presentation system as recited in claim 13, wherein the document is a dynamically generated hypertext markup language (HTML) user interface.

15

16. A data presentation system as recited in claim 13, wherein the document is a dynamically generated hypertext markup language (HTML) control panel user interface.

20

17. A data presentation system as recited in claim 13, wherein:
the first data file is an extensible markup language (XML) data file having
XML data configured to define the document; and
the document is a dynamically generated hypertext markup language
(HTML) user interface incorporating the XML data and the second data.

18. A data presentation system as recited in claim 13, wherein the reference link is a subset of navigable nodes in a navigation architecture, the reference link being configured to intuitively describe the second data.

5 19. A computing system comprising:

a memory;

a processor;

an extensible markup language (XML) data file stored in the memory, the XML data file having control panel UI data configured to define a control panel 10 user interface (UI) for the computing system;

a second data file stored in the memory, the second data file having control panel UI data configured to further define the control panel UI for the computing system;

the XML data file having a reference link to the second data file; and

15 a browser executable on the processor, the browser configured to process the control panel UI data contained in the XML data file and in the second data file to dynamically generate the control panel UI.

20. A computing system as recited in claim 19, further comprising an 20 XML schema stored in the memory, the XML schema configured to define a format of the control panel UI data in the XML data file and in the second data file.

21. A computing system as recited in claim 19, further comprising an extensible stylesheet language (XSL) template stored in the memory wherein the browser has an XSL processor to process the control panel UI data with the XSL template to dynamically generate the control panel UI.

5

22. A computing system as recited in claim 19, wherein the control panel UI is a dynamically generated hypertext markup language (HTML) document.

23. A computing system as recited in claim 19, wherein the browser 10 renders the control panel UI, the control panel UI incorporating the control panel UI data contained in the XML data file and in the second data file.

24. A computing system as recited in claim 19, further comprising a display to render the control panel UI, the control panel UI incorporating for display 15 the control panel UI data contained in the XML data file and in the second data file.

25. A computing system as recited in claim 19, wherein the reference link 20 is a subset of navigable nodes in a navigation architecture, the reference link being configured to intuitively describe the control panel UI data contained in the second data file.

26. A computing system comprising:

- a memory;
- a processor;
- an extensible markup language (XML) data file stored in the memory, the
- 5 XML data file having data configured to define a control panel user interface (UI) for the computing system;
- one or more secondary data files stored in the memory, the one or more secondary data files having control panel UI data configured to further define the control panel UI for the computing system;
- 10 the XML data file having at least one reference link to the one or more secondary data files;
- an XML schema stored in the memory, the XML schema configured to define a format of the control panel UI data contained in the XML data file and in the one or more secondary data files;
- 15 an extensible stylesheet language (XSL) template stored in the memory; and a browser executable on the processor, the browser having an XSL processor to process the control panel UI data with the XSL template to dynamically generate the control panel UI.

20 **27.** A computing system as recited in claim 26, wherein the control panel UI is a dynamically generated hypertext markup language (HTML) document.

28. A computing system as recited in claim 26, wherein the browser renders the control panel UI, the control panel UI incorporating the control panel UI

25 data contained in the XML data file and in the one or more secondary data files.

29. A computing system as recited in claim 26, further comprising a display to render the control panel UI, the control panel UI incorporating for display the control panel UI data contained in the XML data file and in the one or more secondary data files.

5

30. A computing system as recited in claim 26, wherein said at least one reference link is a subset of navigable nodes in a navigation architecture, the said at least one reference link being configured to intuitively describe the control panel UI data contained in the one or more secondary data files.

10

31. A control panel hierarchy comprising:
one or more navigation pages having selectable reference links, the reference links being configured to initiate the dynamic generation of a task page; and
one or more task pages having selectable reference links, the reference links
15 being configured to initiate the dynamic generation of one or more control panel user interfaces (UIs).

32. A control panel hierarchy as recited in claim 31, wherein the one or more navigation pages and the one or more task pages are configured to be rendered
20 as hypertext markup language (HTML) documents.

33. A control panel hierarchy as recited in claim 31, wherein the one or more control panel UIs are configured to be rendered as hypertext markup language (HTML) documents.

25

34. A control panel hierarchy as recited in claim 31, wherein the one or more navigation pages, the one or more task pages, and the one or more control panel UIs are each defined individually by extensible markup language (XML) data contained in one or more XML data files.

5

35. A control panel hierarchy as recited in claim 31, wherein the one or more navigation pages, the one or more task pages, and the one or more control panel UIs are defined collectively by an XML schema.

10 36. A control panel hierarchy as recited in claim 31, wherein the selectable reference links are subsets of navigable nodes in a navigation architecture, the selectable reference links being configured to intuitively describe the one or more task pages and the one or more control panel UIs.

15 37. An extensible markup language (XML) data file stored on a computer-readable medium comprising:

first data to define a document; and
a reference link to invoke a second data file wherein the reference link can be initiated by an application to invoke second data stored in the second data file,
20 the second data coordinating with the first data to further define the document.

38. An XML data file as recited in claim 37, wherein the document is a hypertext markup language (HTML) control panel user interface (UI).

39. An XML data file as recited in claim 37, wherein the reference link is a subset of navigable nodes in a navigation architecture, the reference link being configured to intuitively describe the second data.

5 40. A control panel user interface (UI) presentation system comprising:
 means for invoking an XML data file having XML data configured to define
 a control panel UI and having a reference link to reference a second data file having
 secondary data configured to further define the control panel UI;
 means for loading the XML data and the secondary data into a browser;
10 means for verifying that the XML data and the secondary data is a valid
 representation of an XML schema, the XML schema defining a format of the XML
 data and of the secondary data; and
 means for processing the XML data and the secondary data with an XML
 template to dynamically generate the control panel UI.

15 41. A document presentation system comprising:
 means for invoking a first data file having first data configured to define a
 document and having a reference link to a second data file having second data
 configured to further define the document; and
20 means for processing the first data and the second data to dynamically
 generate the document, the second data invoked for said processing as a result of
 said invoking the first data file.

42. A method for rendering a control panel user interface (UI) comprising:

invoking an XML data file having XML data configured to define a control panel UI and having a reference link to reference a second data file having second data configured to further define the control panel UI;

5 loading the XML data and the second data into a browser;

verifying that the XML data and the second data is a valid representation of an XML schema, the XML schema defining a format of the XML data and of the second data; and

10 processing the XML data and the second data with an XML template to dynamically generate the control panel UI.

43. A method for rendering a control panel user interface (UI) as recited in claim 42 further comprising displaying the control panel UI, the control panel UI 15 incorporating the XML data and the second data for display.

44. A method for rendering a control panel user interface (UI) as recited in claim 42, wherein the reference link is a subset of navigable nodes in a navigation architecture, the reference link being configured to intuitively describe 20 the second data.

45. A computer-readable medium comprising computer executable instructions that, when executed, direct a computing system to perform the method of claim 42.

46. A computer-readable medium comprising computer executable instructions for rendering a document comprising:

invoking a first data file having first data configured to define a document and having a reference link to a second data file having second data configured to further define the document; and

5 processing the first data and the second data to dynamically generate the document, the second data invoked for said processing as a result of said invoking the first data file.

10 47. A computer-readable medium comprising computer executable instructions as recited in claim 46, further comprising displaying the document, the document incorporating the first data and the second data for display.

15 48. A data presentation system comprising:

a data file having data configured to define a document; and

an application having a reference link to the data file, the reference link being configured to intuitively describe the data; and

the application being configured to initiate the reference link to invoke the data such that the document contains the data when generated.

20 49. A data presentation system as recited in claim 48, wherein the document incorporates the data when the document is rendered for display on a display device.

50. A data presentation system as recited in claim 48, wherein the document is a dynamically generated hypertext markup language (HTML) user interface.

5 51. A data presentation system as recited in claim 48, wherein the document is a dynamically generated hypertext markup language (HTML) control panel user interface.

10 52. A data presentation system as recited in claim 48, wherein the data file is an extensible markup language (XML) data file having XML data configured to define the document, and wherein the document is a dynamically generated hypertext markup language (HTML) control panel user interface incorporating the XML data.

15 53. A data presentation system as recited in claim 48, wherein the reference link is a subset of navigable nodes in a navigation architecture.

20 54. An extensible markup language (XML) data structure having data that defines a user interface (UI) stored on a computer readable medium comprising:

an identifier attribute that identifies the XML data structure, the identifier attribute intuitively identifying the data as user interface data;

25 an identifier element that identifies the XML data structure as a navigable node in a navigation architecture, the navigable node labeled as the identifier attribute; and

an extending attribute that defines whether the XML data structure will be an extension to an existing navigable node in the navigation architecture.

55. An extensible markup language (XML) data structure as recited in
5 claim 54, further comprising:

a permissions attribute that defines a user access permission to the XML data structure;

an extensible attribute that designates that the navigable node will not have any children nodes in the navigation architecture;

10 an replaceable attribute that designates that the navigable node cannot be replaced by another navigable node in the navigation architecture;

a title element that defines a name for the navigable node to be displayed on the UI, the name being different than the identifier attribute;

15 a short title element that defines a second name for the navigable node, the second name being a shorter version of the title element name;

a description element that defines a function of the navigable node;

a textual element that defines text for a tool tip, the text configured to be displayed on the UI;

an image element that defines an image to be displayed on the UI;

20 a target element that defines a navigation target to the navigable node;

a search target element that defines a navigable node for a searching UI;

a keywords element that defines one or more words on the UI, the one or more words being identified as search targets for the UI;

a directive element that defines text to be displayed on the UI, the text

25 describing to a user what actions can be initiated on the UI;

- a relational element that defines one or more reference links to one or more related UIs, the one or more reference links being a subset of navigable nodes in a navigation architecture;
- 5 a help element that defines one or more reference links to one or more help topics that are related to the UI, the one or more reference links being a subset of navigable nodes in a navigation architecture;
- a file element that defines a location of hypertext markup language (HTML) data that is configured to be displayed on the UI;
- 10 a second file element that defines a location of a commands file, the commands file defining commands to be displayed on the UI.

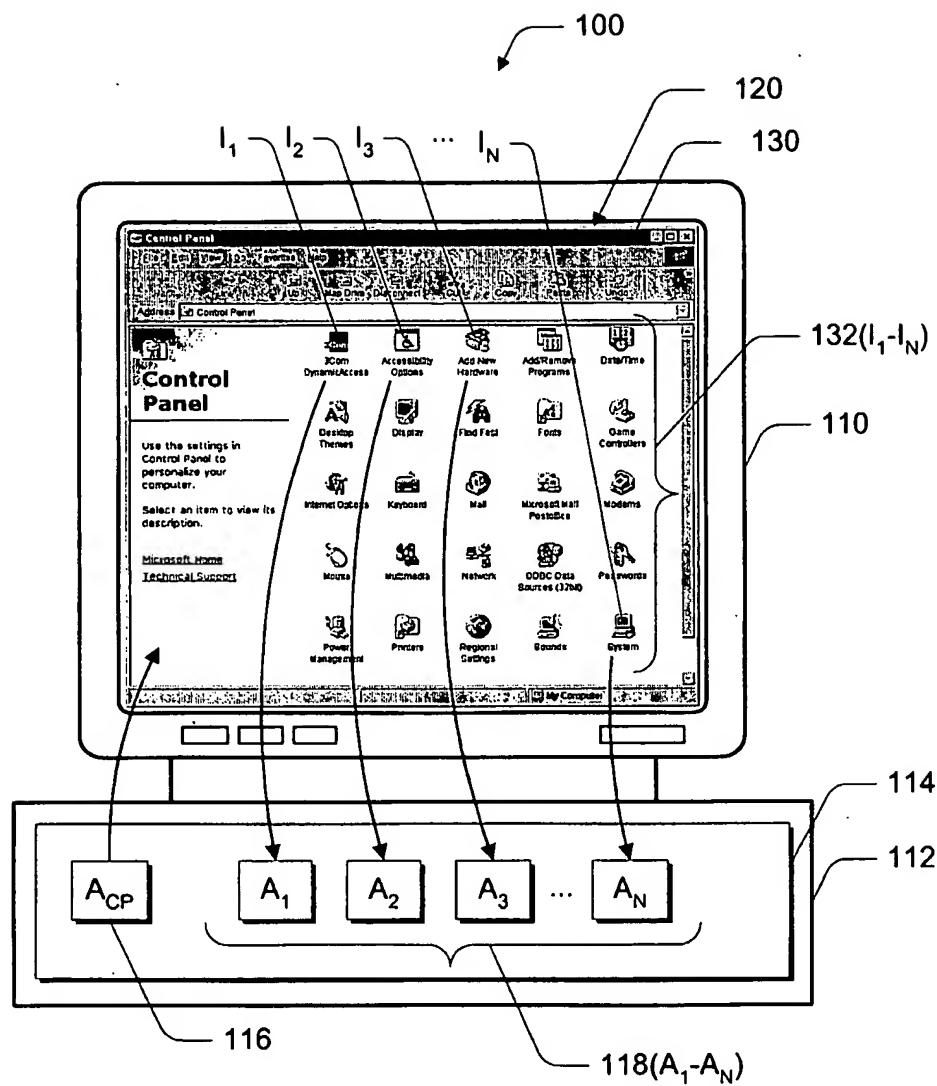
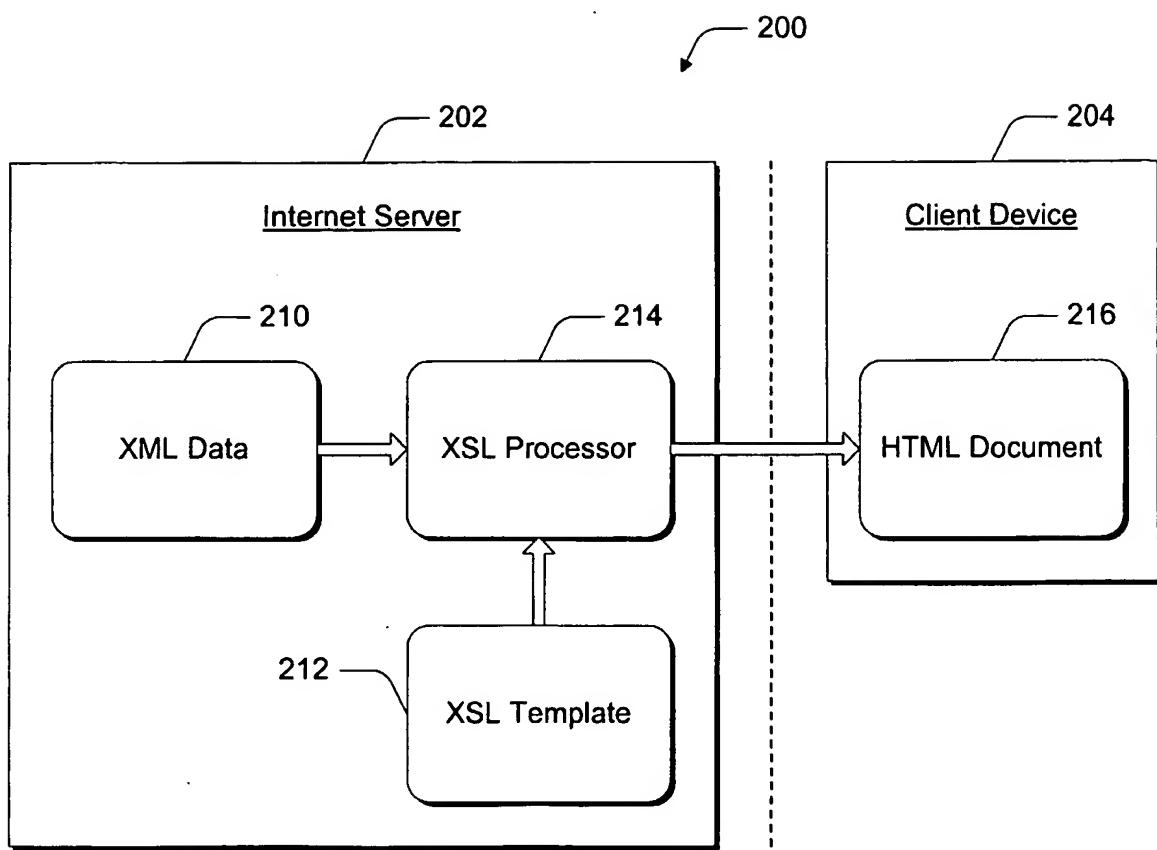


Fig. 1
Prior Art



*Fig. 2
Prior Art*

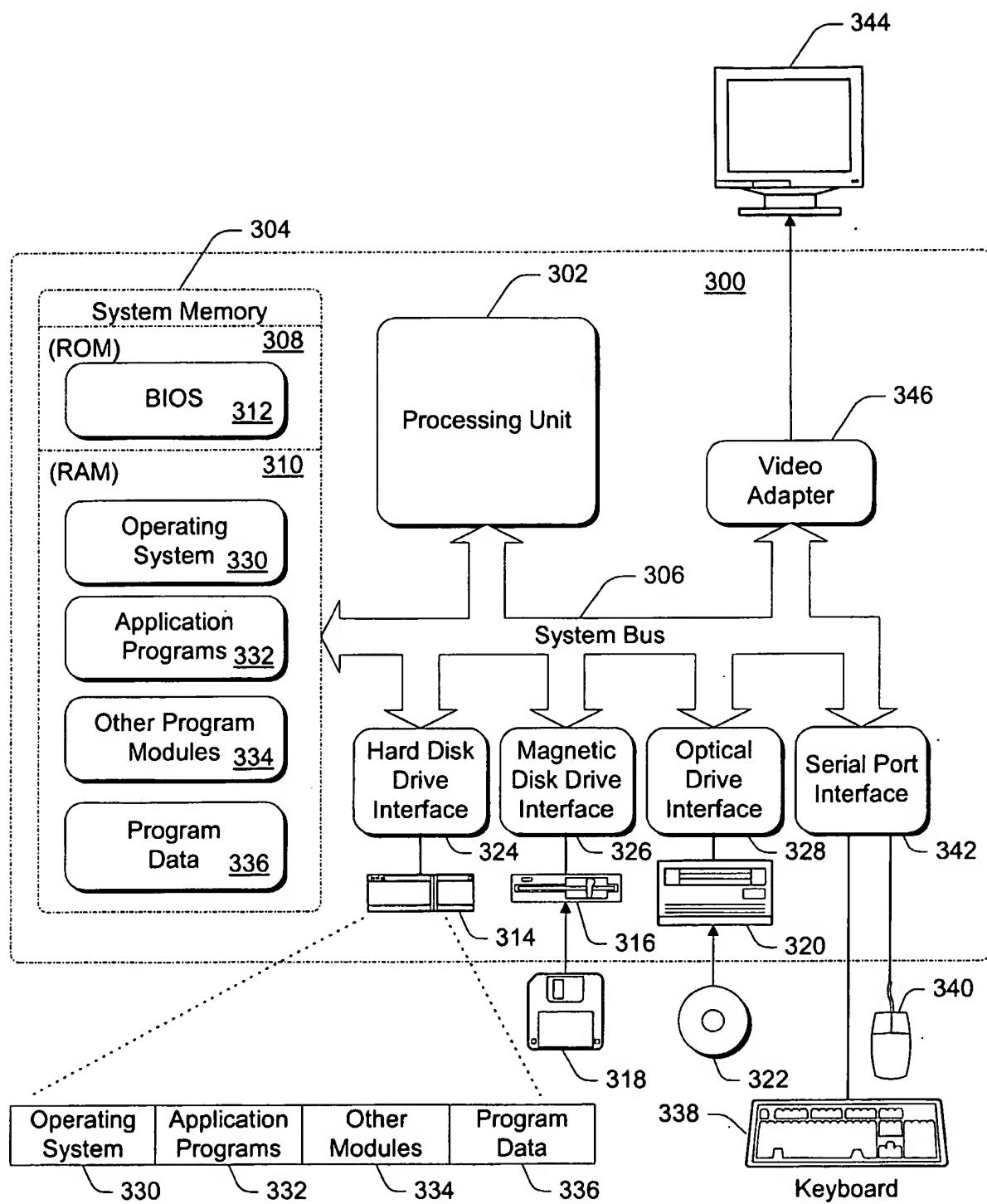
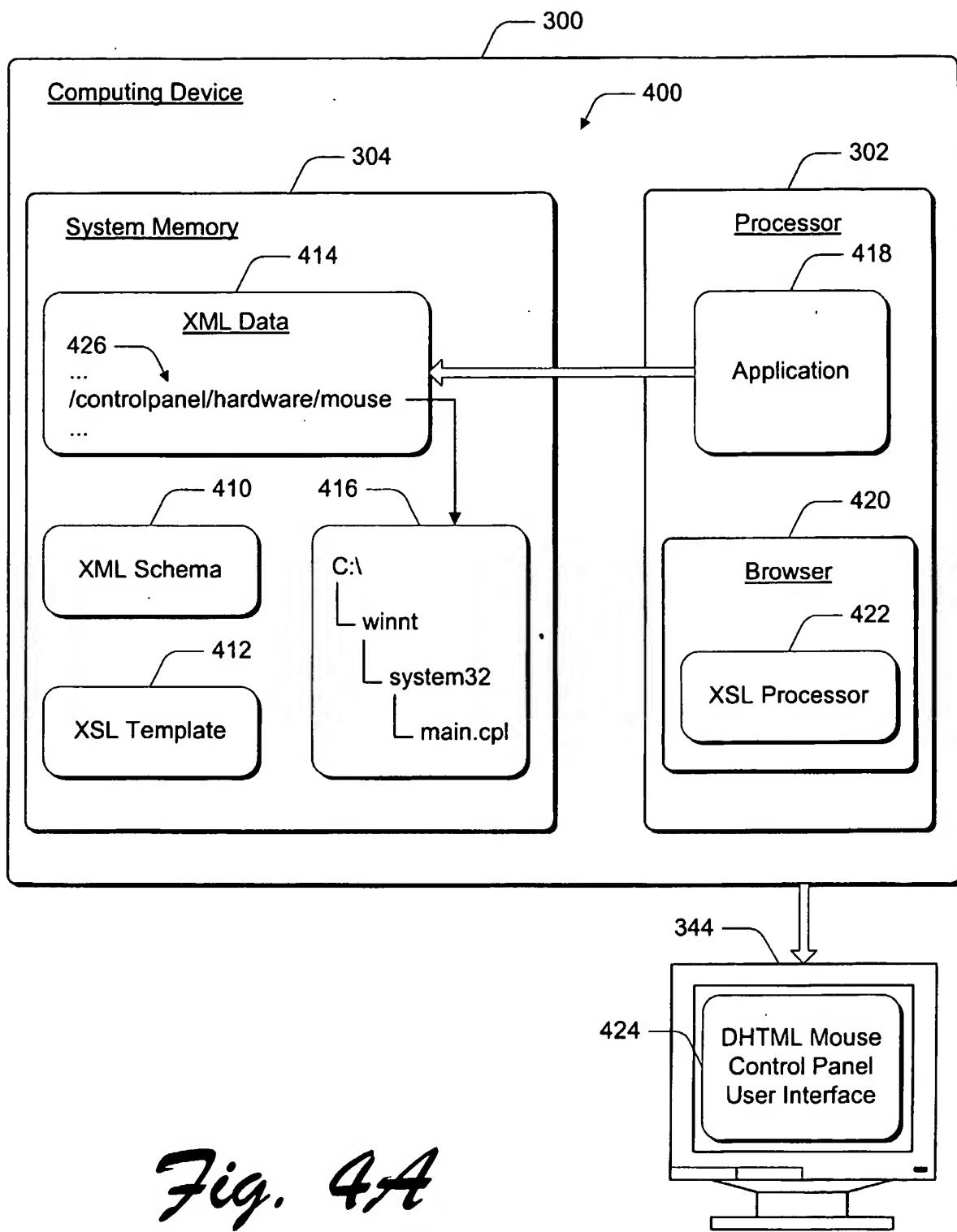


Fig. 3



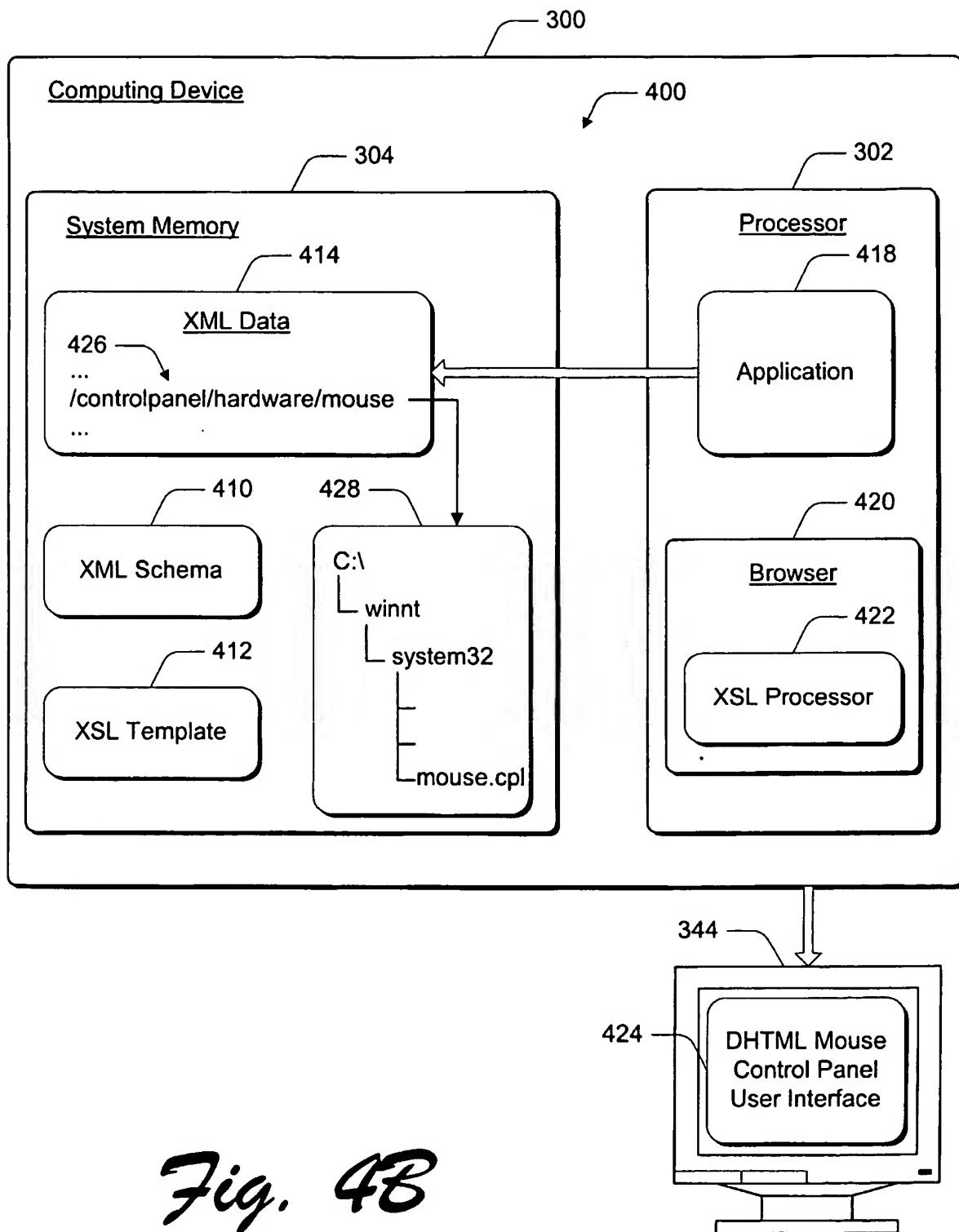
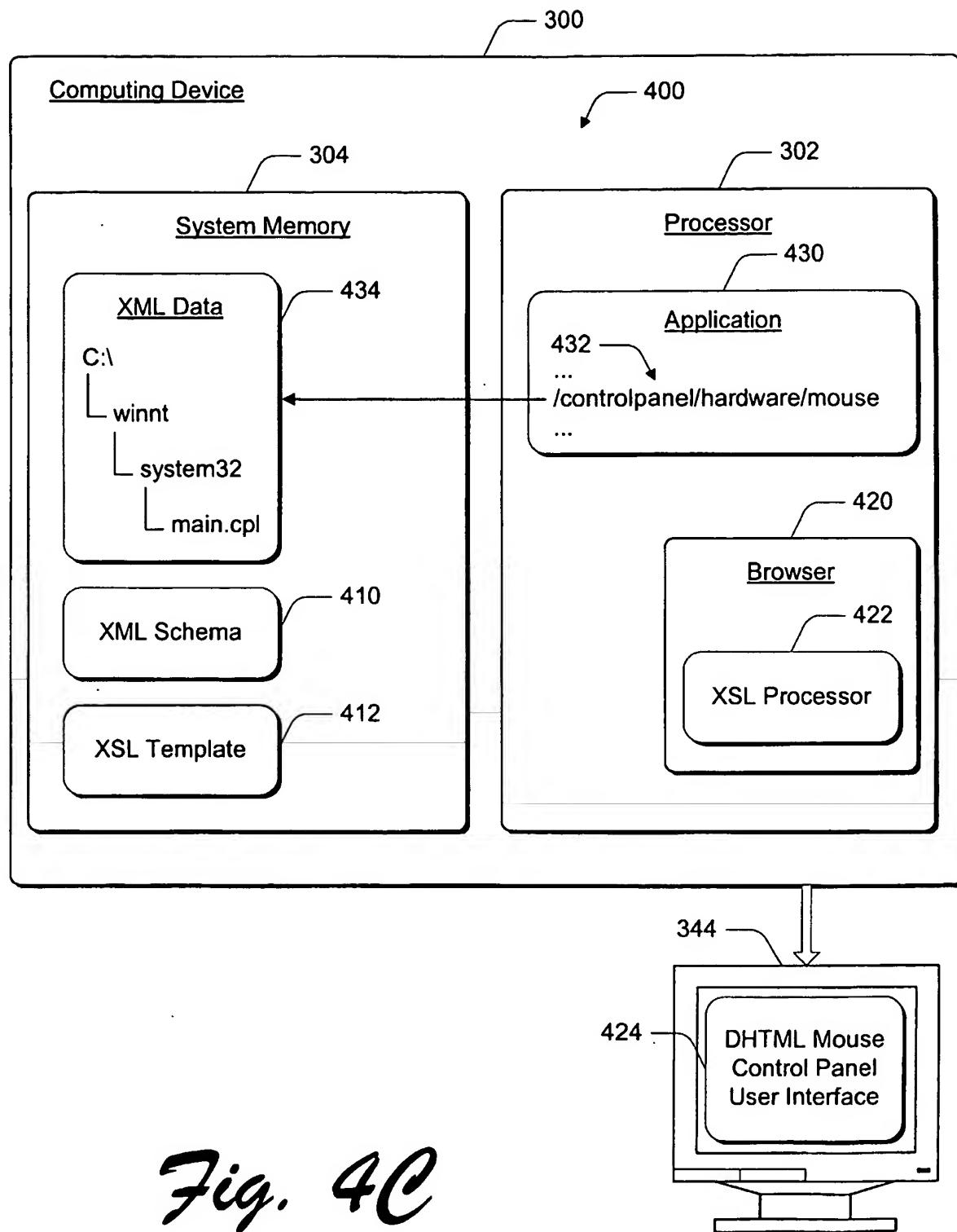


Fig. 4B



7/11

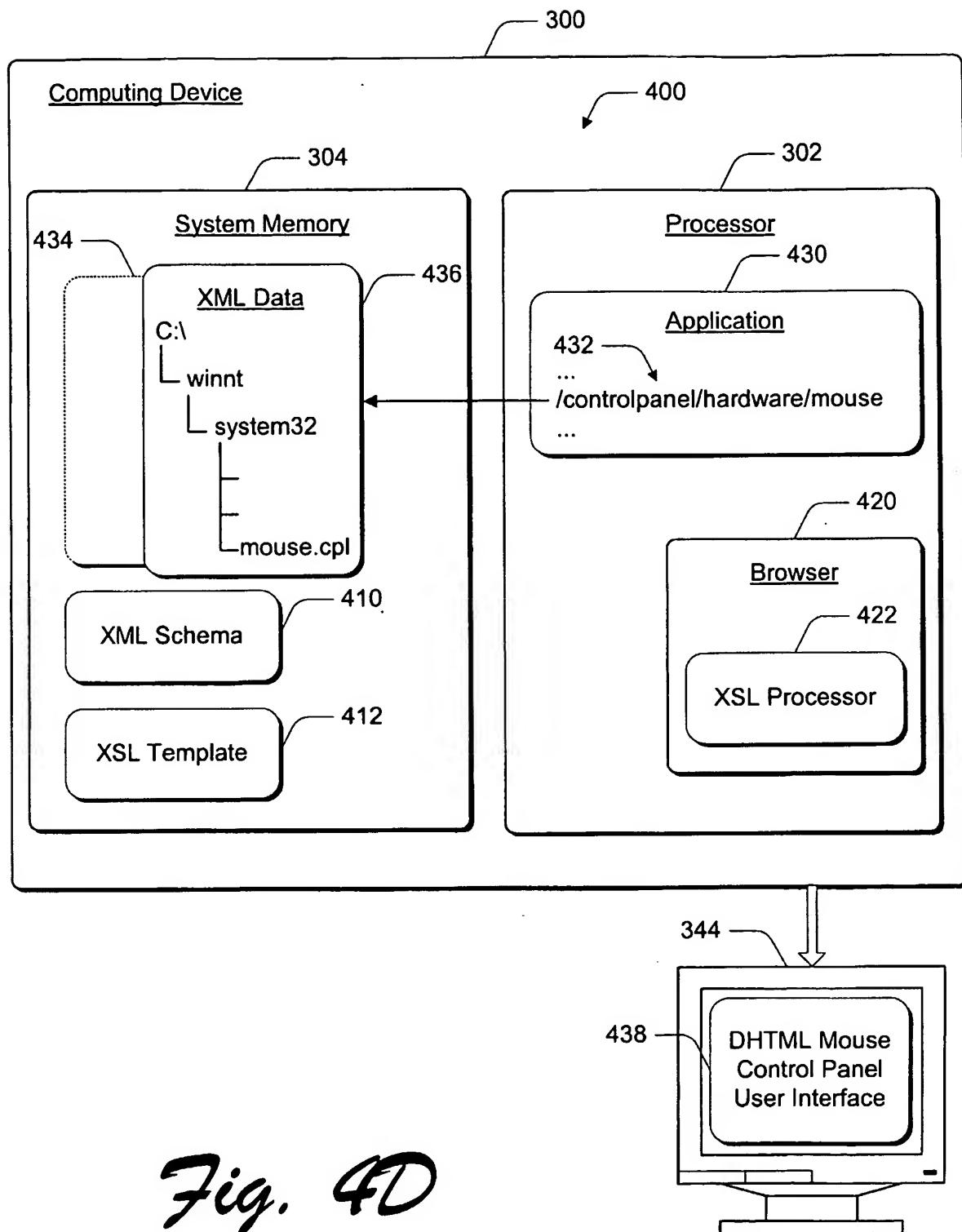


Fig. 4D

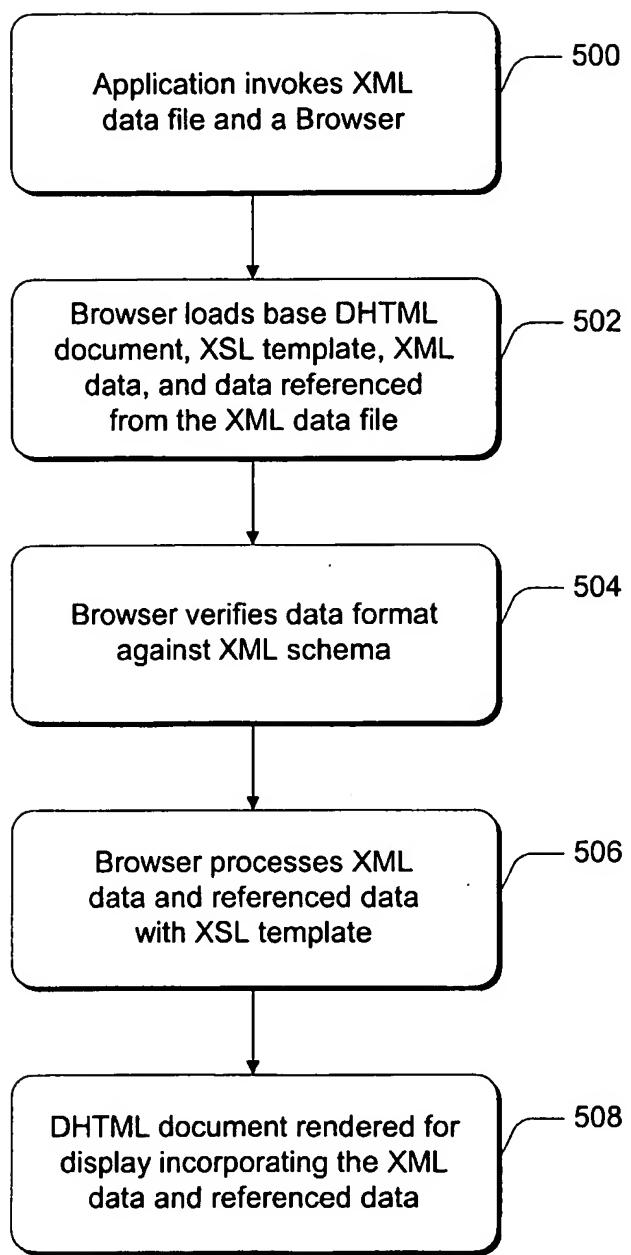


Fig. 5

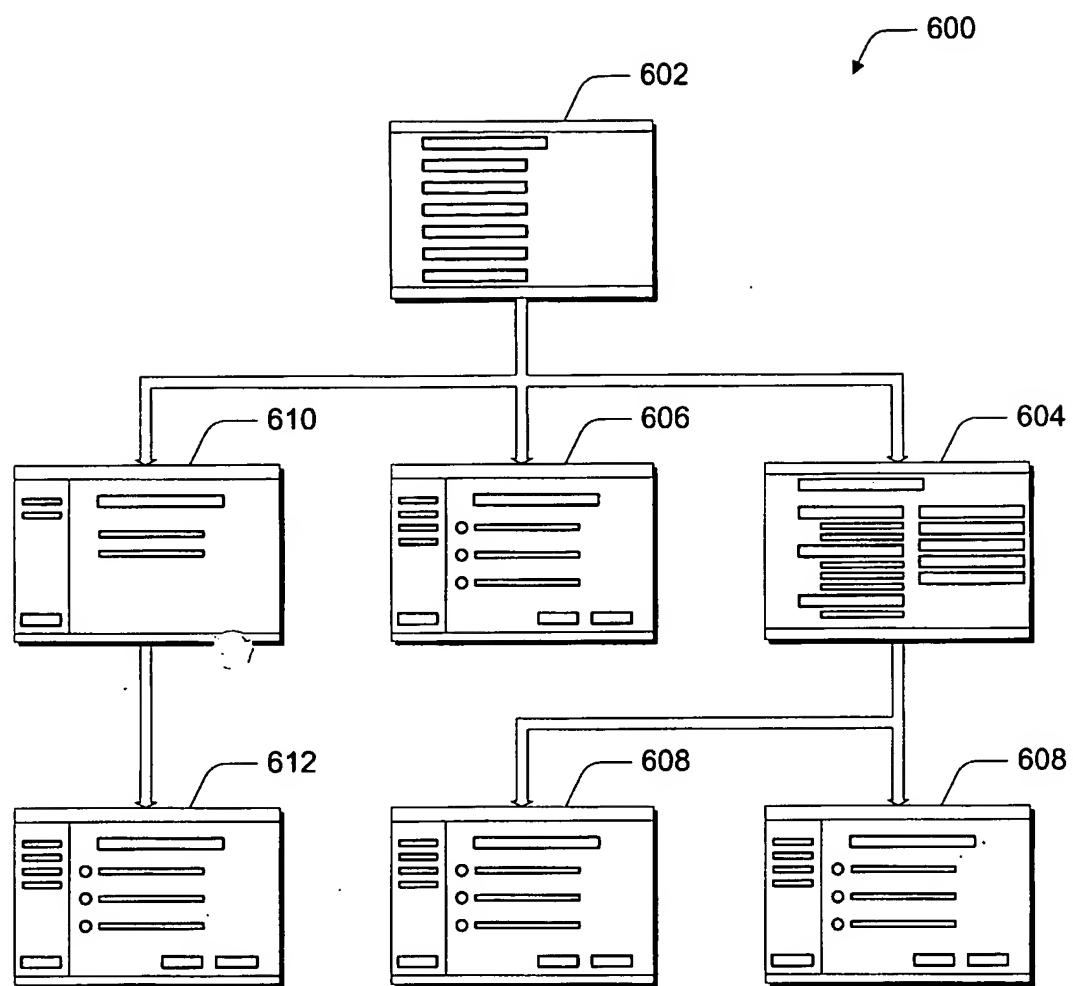


Fig. 6

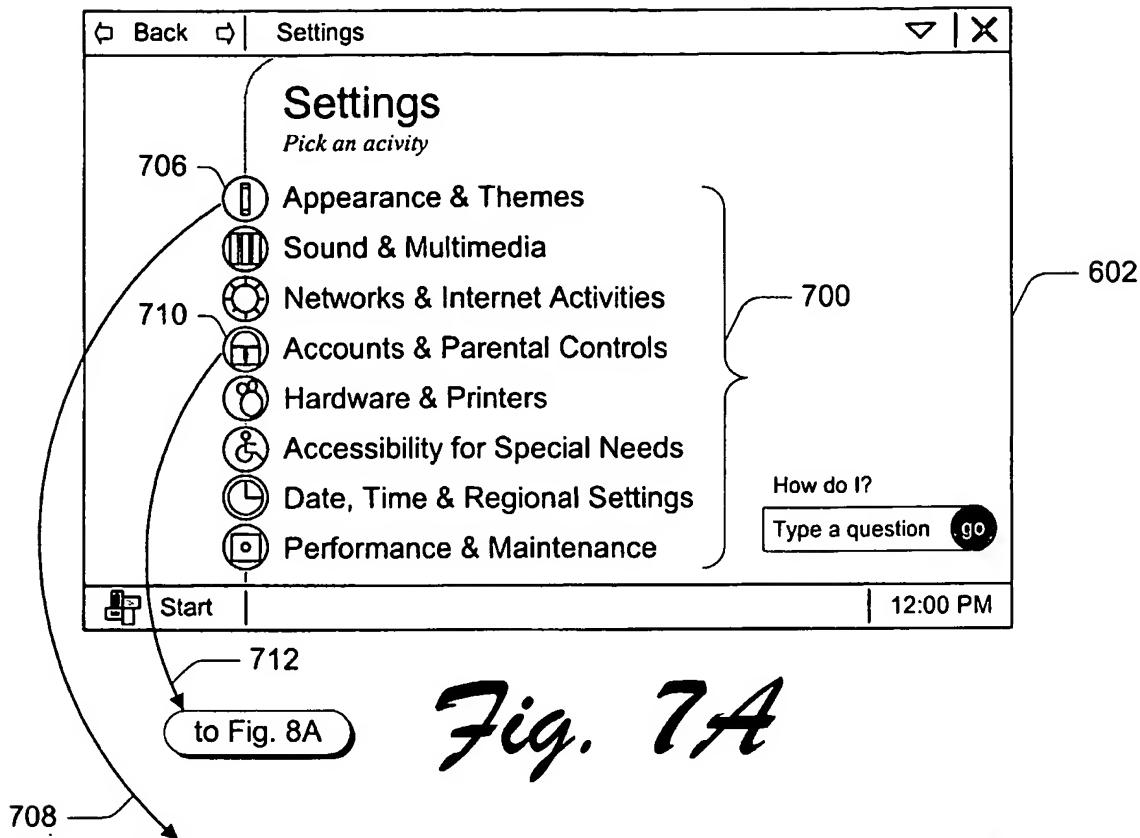


Fig. 7A

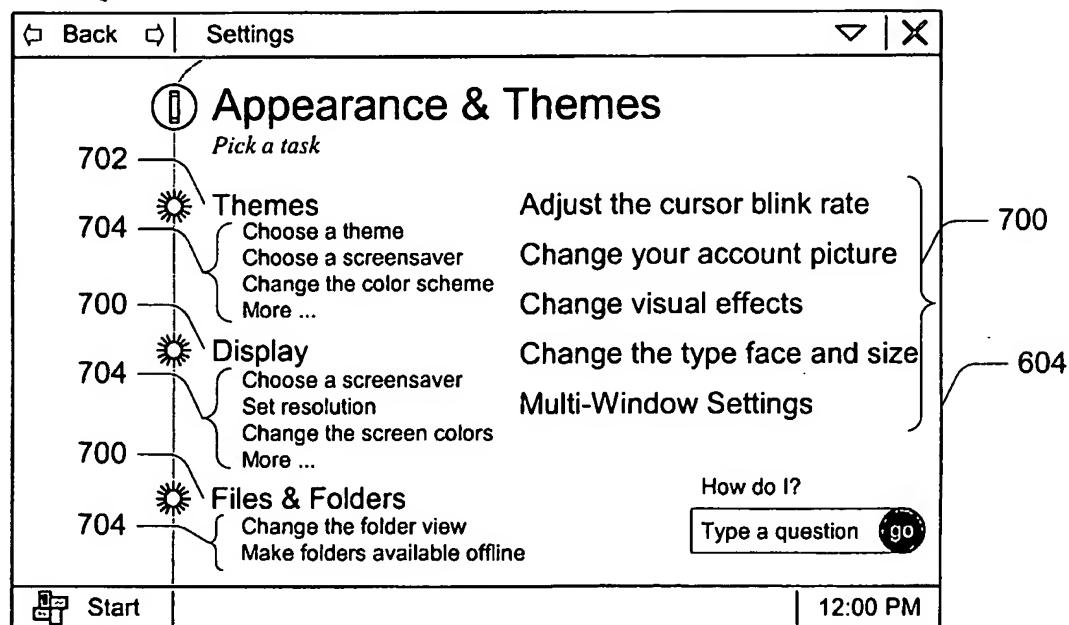
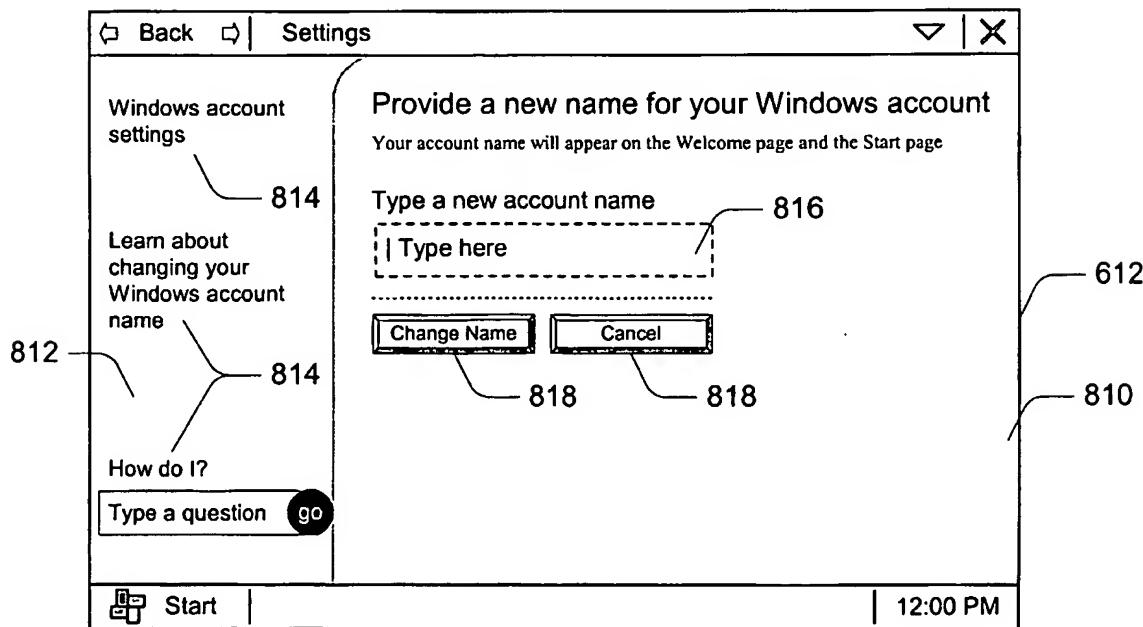
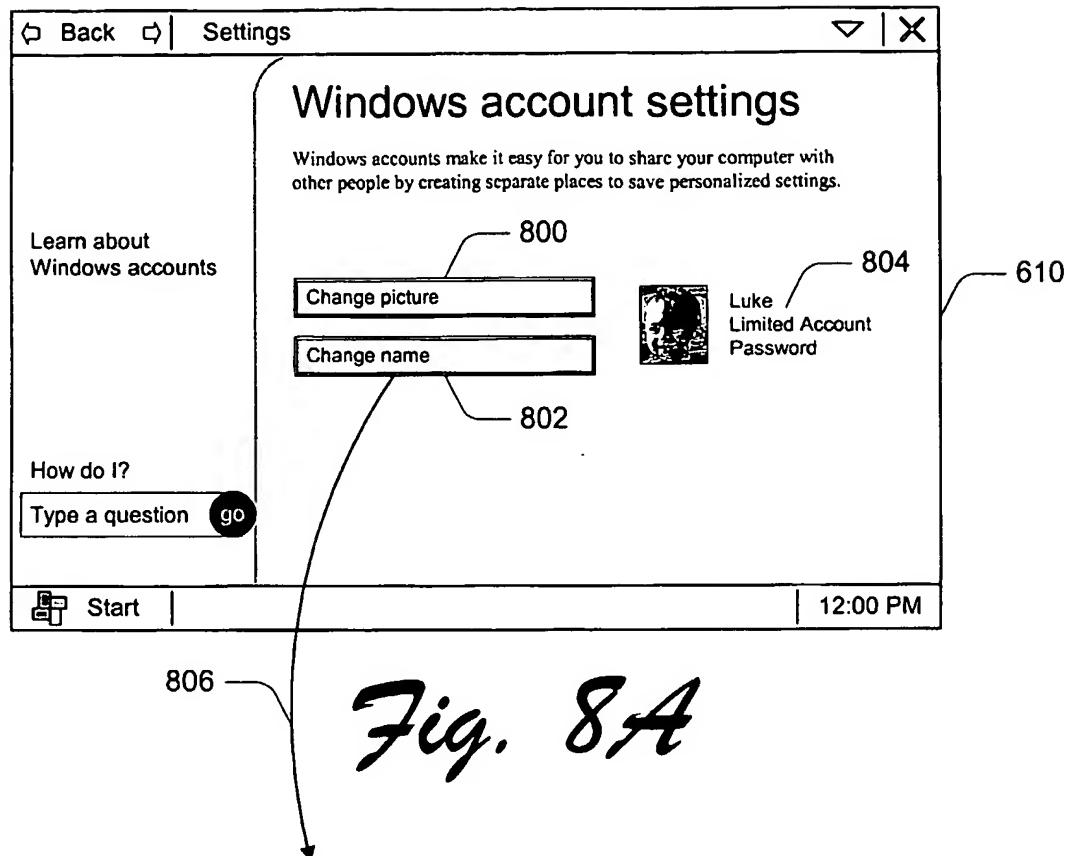


Fig. 7B



INTERNATIONAL SEARCH REPORT

Int'l Application No
PCT/US 00/20723A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, IBM-TDB, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	LUBELL J: "STRUCTURED MARKUP ON THE WEB" MARKUP LANGUAGES, US, MIT PRESS, CAMBRIDGE, MA, vol. 1, no. 3, 1999, pages 7-22, XP000863188 ISSN: 1099-6621 page 18, paragraph 4 -page 20, paragraph 1; figure 5	13-15, 17,18, 37,39, 41, 46-50, 52,53
Y	---	16,38,51
A	---	1-12, 19-36, 40,54,55

 Further documents are listed in the continuation of box C. Patent family members are listed in annex.

* Special categories of cited documents :

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

- *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- *&* document member of the same patent family

Date of the actual completion of the international search

30 January 2001

Date of mailing of the international search report

09/02/2001

Name and mailing address of the ISA

European Patent Office, P.O. Box 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl.
Fax: (+31-70) 340-3016

Authorized officer

Carciofi, A

INTERNATIONAL SEARCH REPORT

Inter
al Application No
PCT/US 00/20723

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	<p>D. ESPOSITO: "A brand new web look for your folders" MSDN ONLINE LIBRARY, 'Online! May 1999 (1999-05), XP002158800 Retrieved from the Internet: <URL:http://msdn.microsoft.com/library/tech/webview.htm> 'retrieved on 2001-01-24! page 1, line 1 - line 10; figures 3,6 page 1, line 25 - line 29 page 3, line 4 - line 8 page 4, line 3 - line 12 page 4, line 27 -page 5, line 32 page 7, line 24 - line 36 page 8, line 33 -page 11, line 16</p>	1-12, 19-36, 40, 42-45
A	---	54, 55
Y	<p>P. DILASCIA: "C++ Q&A" MICROSOFT SYSTEMS JOURNAL, 'Online! November 1999 (1999-11), XP002158801 Retrieved from the Internet: <URL:http://www.microsoft.com/msj/defaultframe.asp?page=/msj/1199/c++/c++1199.htm> 'retrieved on 2001-01-24! page 1, paragraph 4; figure 2 page 2, paragraph 1 page 6, paragraph 2 - paragraph 3</p>	1-12, 19-36, 40, 42-45
Y	<p>S. SANDOLO, A. SAPPIA: "JAD System Architecture" JAVA ADMINISTRATION SYSTEM DOCUMENTATION, 'Online! 30 August 1997 (1997-08-30), XP002158802 Retrieved from the Internet: <URL:http://server-esng.dibe.unige.it/Projects/ClosedProjects/JAD/reserved/JadSystemArchitecture.html> 'retrieved on 2001-01-24! page 1, paragraph 1 - paragraph 2; figure 1 page 2, paragraph 7 -page 4, paragraph 2 page 4, last paragraph</p>	16, 38, 51
A	---	1, 8, 19, 26, 31, 40, 42, 45
A	<p>"DISK DRIVE WITH EMBEDDED HYPER-TEXT MARKUP LANGUAGE SERVER" IBM TECHNICAL DISCLOSURE BULLETIN, US, IBM CORP. NEW YORK, vol. 38, no. 12, 1 December 1995 (1995-12-01), page 479 XP000588211 ISSN: 0018-8689 the whole document</p>	1, 8, 16, 19, 26, 31, 38, 40, 42, 45, 51
	---	-/-

INTERNATIONAL SEARCH REPORT

Inte
al Application No
PCT/US 00/20723

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>K. SALL: "Doing it with XML, part 1" WEB DEVELOPER'S VIRTUAL LIBRARY - TUTORIALS, 'Online! April 1999 (1999-04), XP002158803 Retrieved from the Internet: <URL:http://wdvl.com/Authoring/Languages/XML/Tutorials/DoingIt/index.html> 'retrieved on 2001-01-24! page 18 -page 23 ---</p>	2,5-12, 20, 23-30, 35,36, 40,42, 45,54,55
A	<p>S. SOL: "Introduction to XML For Web Developers - Part Three: the DTD" WEB DEVELOPER'S VIRTUAL LIBRARY - TUTORIALS, 'Online! 3 May 1999 (1999-05-03), XP002158804 Retrieved from the Internet: <URL:http://wdvl.com/Authoring/Languages/XML/Tutorials/Intro/index3.html> 'retrieved on 2001-01-24! page 3 -page 5 page 15 -----</p>	2,5-12, 20, 23-30, 35,36, 40,42, 45,54,55